HHS Public Access

Author manuscript

KDD. Author manuscript; available in PMC 2015 April 29.

Published in final edited form as:

KDD. 2012 ; 2012: 280–288. doi:10.1145/2339530.2339578.

Mining Recent Temporal Patterns for Event Detection in Multivariate Time Series Data

Iyad Batal,
Dept. of Computer Science, University of Pittsburgh, iyad@cs.pitt.edu

Dmitriy Fradkin,
Siemens Corporate Research, dmitriy.fradkin@siemens.com

James Harrison,
Dept. of Public Health Sciences, University of Virginia, james.harrison@virginia.edu

Fabian Moerchen, and
Siemens Corporate Research, fabian.moerchen@siemens.com

Milos Hauskrecht
Dept. of Computer Science, University of Pittsburgh, milos@cs.pitt.edu

Abstract

Improving the performance of classifiers using pattern mining techniques has been an active topic of data mining research. In this work we introduce the recent temporal pattern mining framework for finding predictive patterns for monitoring and event detection problems in complex multivariate time series data. This framework first converts time series into time-interval sequences of temporal abstractions. It then constructs more complex temporal patterns backwards in time using temporal operators. We apply our framework to health care data of 13,558 diabetic patients and show its benefits by efficiently finding useful patterns for detecting and diagnosing adverse medical conditions that are associated with diabetes.

Keywords

Temporal Pattern Mining; Temporal Abstractions; Time-interval Patterns; Event Detection; Patient Classification

1. INTRODUCTION

Advances in data collection and data storage technologies led to emergence of complex temporal datasets, where the data instances are traces of complex behaviors characterized by time series of multiple variables. Designing algorithms capable of learning classification models from such data is one of the most challenging topics of data mining research.

The majority of existing classification methods that work with temporal data [12, 6, 4, 24, 8, 28] assume that each data instance (represented by a single or multiple time series) is associated with a single class label that affects its *entire* behavior. That is, they assume that all temporal observations are equally useful for classification.

However, the above assumption is not the best when considering monitoring and event detection problems. In this case, the class label denotes an event that is associated with a specific time point (or a time interval) in the instance, not necessarily in the entire instance. The goal is to learn a model that can accurately identify the occurrence of events in unlabeled instances (a monitoring task). Examples of such problems are the detection of adverse medical events (e.g. drug toxicity) in clinical data [10], detection of the equipment malfunction [9], fraud detection [23], environmental monitoring [16], intrusion detection [7] and others.

Given that class labels are associated with specific time points (or time intervals), each instance can be annotated with multiple labels[1]. Consequently, the context in which the classification is made is often local and affected by the most *recent* behavior of the monitored instances.

The focus of this paper is to develop a pattern mining technique that takes into account the local nature of decisions for monitoring and event detection problems. We propose the ***Recent Temporal Pattern* (*RTP*)** mining framework, which mines frequent temporal patterns backward in time, starting from patterns related to the most recent observations. Applying this technique, temporal patterns that extend far into the past are likely to have low support in the data and hence would not be considered for classification. Incorporating the concept of recency in temporal pattern mining is a new research direction that, to the best of our knowledge, has not been previously explored in the pattern mining literature.

We study our *RTP* mining approach by analyzing temporal data encountered in Electronic Health Record (EHR) systems. In EHR data, each record (data instance) consists of multiple time series of clinical variables collected for a specific patient, such as laboratory test results and medication orders. The record may also provide information about patient's diseases and adverse medical events over time. Our objective is to learn classification models that can accurately detect adverse events and apply it to monitor future patients.

The task of temporal modeling for EHR data is challenging because the data are *multivariate* and the time series for clinical variables are *irregularly sampled in time* (measured asynchronously at different time moments). Therefore, most existing times series classification methods [6, 24], time series similarity measures [29, 20] and time series feature extraction methods [4, 13] cannot be directly applied on the raw EHR data.

This paper proposes a temporal pattern mining approach that can handle complex data such as EHR. The key step for this approach is defining a language that can adequately represent the temporal dimension of the data. Our approach relies on 1) temporal abstractions [21] to

[1]In the clinical domain, a patient may be healthy at first, then develop an adverse medical condition, then be cured and so on.

convert numeric time series variables to *time-interval* sequences and 2) temporal relations [3] to represent temporal interactions among the variables. For example, this allows us to define complex *temporal patterns* (time-interval patterns) such as "the administration of heparin precedes a decreasing trend in platelet counts".

After defining patterns from temporally abstracted data, we need to design an efficient mining algorithm for finding patterns that are useful for event detection. Mining time-interval data is a relatively young research field that extends sequential pattern mining [2, 31, 18, 30] to the more complex case of time-interval pattern mining[2]. Most existing methods mine frequent patterns in an unsupervised way in order to find temporal association rules [22, 11, 17, 14, 26, 27, 15]. Our objective is different because we are interested in mining temporal patterns that are potentially important for the event detection task. To address this, we present an efficient algorithm for mining *RTPs* (see above) from time-interval data.

We test and demonstrate the usefulness of our framework on real-world EHR data collected for 13,558 diabetes patients. Our task is to learn classification models that can correctly diagnose disorders associated with diabetes, such as cardiological, renal or neurological disorders. We first show that incorporating the temporal dimension is beneficial for this task. In addition, we show the following advantages of our framework:

1. *RTP* mining focuses the search on temporal patterns that are potentially more useful for classification.
2. The number of frequent *RTPs* is much smaller than the number of frequent temporal patterns. This can facilitate the process of reviewing and validating the mined patterns by human experts.
3. Our mining algorithm is much more efficient than other temporal pattern mining approaches and it can scale up much better to large datasets.

2. PROBLEM DEFINITION

Let $D=\{<x_i, y_i>\}_{i=1}^{n}$ be a training dataset such that $x_i \in X$ is a multivariate temporal instance up to some time t_i and $y_i \in Y$ is a class label associated with x_i at time t_i. The objective is to learn a function f: $X \rightarrow Y$ that can label unlabeled instances. This general setting is applicable to different monitoring and event detection problems, such as the ones described in [23, 7, 9, 16].

In this work, we test our method on data from electronic health records (EHR), hence we will use the EHR application as example throughout the paper. For this task, every data instance x_i is a record for a specific patient up to time t_i and the class label y_i denotes whether or not this patient is diagnosed with an adverse medical condition (e.g., renal failure) at t_i. Figure 1 shows a graphical illustration of an EHR instance with 3 clinical

[2]Sequential pattern mining is a special case of time-interval pattern mining, in which all intervals are instantaneous (with zero durations).

temporal variables. The objective is to learn a classifier that can predict well the studied medical condition and apply it to monitor future patients.

Learning the classifier directly from EHR data is very difficult because the instances consist of multiple irregularly sampled time series of different length. Therefore, we want to apply a space transformation $\psi: X \rightarrow X'$ that maps each instance x_i to a fixed-size feature vector x_i', while preserving the predictive temporal characteristics of x_i as much as possible.

One approach to define ψ is to represent the data using a *predefined* set of features and their values (a *static transformation*) as in [10]. Examples of such features are "most recent platelet measurement", "most recent platelet trend", "maximum hemoglobin measurement", etc. Our approach is different and we *learn* transformation ψ from the data using temporal pattern mining (a *dynamic transformation*). This is done by applying the following steps:

1. Convert the numeric time series variables into time interval sequences using temporal abstraction.
2. Mine recent temporal patterns from the time interval data.
3. Transform each instance x_i into a binary indictor vector x_i' using the patterns obtained in step 2.

After applying transformation ψ, we can use a standard machine learning method (e.g. support vector machines, decision tree, or logistic regression) on $\left\{<x_i', y_i>\right\}_{i=1}^{n}$ to learn function *f*.

In the following, we explain in details each of these steps.

3. TEMPORAL ABSTRACTION PATTERNS

3.1 Temporal Abstraction

The goal of **temporal abstraction** [21] is to transform the numeric time series variables to a high-level qualitative form. More specifically, each clinical variable (e.g., series of white blood cell counts) is transformed into an *interval-based* representation $\langle v_1[s_1, e_1], \ldots, v_n[s_n, e_n]\rangle$, where $v_i \in \Sigma$ is an abstraction that holds from time s_i to time e_i and Σ is the **abstraction alphabet** that represents a finite set of permitted abstractions.

For the EHR data, we segment all laboratory variables based on their values into the following abstract states: *Very Low* (*VL*), *low* (*L*), *Normal* (*N*), *High* (*H*) and *Very High* (*VH*), i.e., $\Sigma = \{VL, L, N, H, VH\}$. We use the 10th, 25th, 75th and 90th percentiles of the lab values to define these 5 states: a value below the 10th percentile is *very low* (*VL*), a value between the 10th and 25th percentiles is *low* (*L*), and so on.

3.2 Multivariate State Sequences

Let a **state** be an *abstraction for a specific variable*. We denote a state *S* by a pair (*F*, *V*), where *F* is a temporal variable and $V \in \Sigma$ is an abstraction value. Let a **state interval** be a *state that holds during an interval*. We denote a state interval *E* by a 4-tuple (*F*, *V*, *s*, *e*),

where F is a temporal variable, $V \in \Sigma$ is an abstraction value, and s and e are the start time and end time (respectively) of the state interval ($E.s \leq E.e$)[3]. For example, assuming the time granularity is days, (*glucose*, *H*, 5, 10) represents high glucose values from day 5 to day 10.

After abstracting all time series variables, we represent every instance x_i in the database D as a **Multivariate State Sequence (MSS)** Z_i. Let $\boldsymbol{Z_i.end}$ denote the end time of the instance.

For notational convenience, we represent an MSS Z_i as a series of state intervals that are sorted according to their start times[4]:

$$Z_i = \langle E_1, E_2, \ldots, E_l \rangle \quad : E_j.s \leq E_{j+1}.s : j \in \{1, \ldots, l-1\}$$

Note that we do not require $E_j.e$ to be less than $E_{j+1}.s$ because the state intervals are obtained from different temporal variables and their intervals may overlap.

Example 1. *Figure 2 shows an MSS Z_i with two temporal variables: creatinine (C) and glucose (G). Assuming the time granularity is days, this MSS represents 24 days of the patient's record ($Z_i.end = 24$). For instance, we can see that the creatinine values are normal from day 2 until day 14, then become high from day 15 until day 24. We represent Z_i as:* $\langle E_1 = (G, H, 1, 5), E_2 = (C, N, 2, 14), E_3 = (G, N, 6, 9), E_4 = (G, H, 10, 13), E_5 = (C, H, 15, 24), E_6 = (G, VH, 16, 23) \rangle$.

3.3 Temporal Relations

The temporal relation between two instantaneous events (time points) can be easily described using three relations: *before*, *at the same time* and *after*. However, when the events have time durations (state intervals), the relations become more complex. Allen [3] described the temporal relation between two state intervals using 13 possible relations (Figure 3). But it suffices to use the following 7 relations: *before*, *meets*, *overlaps*, *is-finished-by*, *contains*, *starts* and *equals* because the other relations are simply their inverses. Allen's relations have been introduced in artificial intelligence for temporal reasoning and have been used later in the fairly young research of time interval data mining [22, 11, 17, 26, 15].

As we can see, most of these relations require equality of one or two of the intervals' end points. That is, there is only a slight difference between *overlaps*, *is-finished-by*, *contains*, *starts* and *equals* relations. When the time information in the data is noisy, which is the case for EHR data, using Allen's relations may cause the problem of pattern fragmentation[5] [14].

Therefore, we opt to use only two temporal relations: ***before* (*b*)** and ***co-occurs* (*c*)**, which we define as follows:

Given two state intervals E_i and E_j:

[3] If $E.s = E.e$, state interval E corresponds to a time point.
[4] If two state intervals have the same start time, we sort them by their end time. If they also have the same end time, we sort them by lexical order (see [11]).
[5] Having many different temporal patterns that describe a very similar situation in the data.

- E_i is ***before*** E_j, denoted as $\boldsymbol{b(E_i, E_j)}$, if $E_i.e < E_j.s$ (same as Allen's *before*).
- E_i ***co-occurs*** with E_j, denoted as $\boldsymbol{c(E_i, E_j)}$, if $E_i.s \leq E_j.s \leq E_i.e$. That is, E_i starts before E_j and there is a nonempty time period where both E_i and E_j occur. Note that this relation covers the following Allen's relations: *meets*, *overlaps*, *is-finished-by*, *contains*, *starts* and *equals*.

3.4 Temporal Patterns

In order to obtain temporal descriptions of the data, basic states are combined using temporal relations to form temporal patterns (time interval patterns). In the previous section, we defined the relation between two states to be either *before* (*b*) or *co-occurs* (*c*). In order to define relations between *k* states, we use Höppner's representation of temporal patterns [11].

Definition 1. *(**Temporal Pattern**) A temporal pattern is defined as* $P = (\langle S_1, \ldots, S_k \rangle, R)$ *where* S_i *is the* i^{th} *state of the pattern and R is an upper triangular matrix that defines the temporal relations between each state and all of its following states:* $i \in \{1, \ldots, k-1\} \wedge j \in \{i+1, \ldots, k\}$*:* $R_{i,j} \in \{b, c\}$ *specifies the relation between* S_i *and* S_j.

The size of a temporal pattern *P* is the number of states it contains. If *P* contains *k* states, we say that *P* is a *k-pattern*. Hence, a single state is a *1-pattern* (a singleton). We also denote the space of all temporal patterns of arbitrary size by ***TP***.

Figure 4 shows a graphical representation of a *4-pattern* $\langle S_1 = (C, H), S_2 = (G, N), S_3 = (B, H), S_4 = (G, H) \rangle$, where the states are abstractions of temporal variables creatinine (C), glucose (G) and BUN (Blood Urea Nitrogen) (B). The half matrix on the right represents the temporal relations between every state and the states that follow it. For example, the first state S_1 *co-occurs* with the third state S_3: $R_{1,3} = c$.

Definition 2. *Given an MSS* $Z = \langle E_1, E_2, \ldots, E_l \rangle$ *and a temporal pattern* $P = (\langle S_1, \ldots, S_k \rangle, R)$*, we say that Z **contains** P, denoted as* $\boldsymbol{P \in Z}$*, if there is an injective mapping* π *from the states of P to the state intervals of Z such that:*

$$\forall i \in \{1, \ldots, k\}: \quad S_i F = E_{\pi(i)}.F \wedge S_i.V = E_{\pi(i)}.V$$
$$\forall i \in \{1, \ldots, k-1\} \wedge j \in \{i+1, \ldots, k\}: \quad R_{ij}\left(E_{\pi(i)}, E_{\pi(j)}\right)$$

The definition says that checking whether an MSS contains a *k-pattern* requires: 1) matching all *k* states of the pattern and 2) checking that all $k(k-1)/2$ temporal relations are satisfied. As an example, the MSS in Figure 2 contains the temporal pattern $P = (\langle (C, N), (G, N) \rangle, R_{1,2} = c)$ (normal creatinine *co-occurs* with normal glucose). To improve readability, we usually write *2-patterns* of the form $(\langle S_1, S_2 \rangle, R_{1,2})$ simply as $S_1\ R_{1,2}\ S_2$. That is, we can write $P = (C, N)\ c\ (G, N)$.

4. MINING RECENT TEMPORAL PATTERNS

4.1 Recent Temporal Patterns

In the event detection setting, each training temporal instance x_i (e.g. an electronic health record) is associated with class label y_i at time t_i (e.g. whether or not a medical condition is detected). Consequently, recent measurements of the variables of x_i (close to t_i) are usually more predictive than distant measurements, as was shown in [25] for clinical data. In the following, we present the definitions of recent state intervals and recent temporal patterns.

Definition 3. *Given an MSS $Z = \langle E_1, E_2, \ldots, E_l \rangle$ and a maximum gap parameter g, we say that $E_j \in Z$ is a* ***recent state interval*** *in Z, denoted as $\mathbf{r_g(E_j, Z)}$, if* ***any*** *of the following two conditions are satisfied:*

- $\nexists E_k \in Z: E_k.F = E_j.F \wedge k > j$
- $Z.end - E_j.e \leq g$

The first condition is satisfied if E_j is the most recent state interval in its variable ($E_j.F$) and the second condition is satisfied if E_j is less than g time units away from the end of the MSS ($Z.end$). Note that if $g = \infty$, any $E_j \in Z$ is considered to be recent.

Definition 4. *(**RTP**) Given an MSS $Z = \langle E_1, E_2, \ldots, E_l \rangle$ and a maximum gap parameter g, we say that temporal pattern $P = (\langle S_1, \ldots, S_k \rangle, R)$ is a* ***Recent Temporal Pattern (RTP)*** *in Z, denoted as $\mathbf{R_g(P, Z)}$, if* ***all*** *the following conditions are satisfied:*

1. *$P \in Z$ with a mapping π from the states of P to the state intervals of Z*
2. *S_k matches a recent state interval in Z: $r_g(E_{\pi(k)}, Z)$*
3. *$\forall i \in \{1, \ldots, k-1\}$, S_i and S_{i+1} match state intervals not more than g away from each other: $E_{\pi(i+1)}.s - E_{\pi(i)}.e \leq g$*

The definition says that in order for temporal pattern P to be an *RTP* in MSS Z, 1) P should be contained in Z (Definition 2), 2) the last state of P should map to a recent state interval in Z (Definition 3), and 3) any pair of consecutive states in P should map to state intervals that are "close to each other". This forces the pattern to be close to the end of Z and to have a limited temporal extension in the past. Note that g is a parameter that specifies the restrictiveness of the *RTP* definition. If $g = \infty$, any pattern $P \in Z$ would be considered to be an *RTP* in Z. When an *RTP* contains k states, we call it a *k-RTP*.

Example 2. *Let Z_i be the MSS in* Figure 2 *and let the maximum gap parameter be $g = 5$ days. Temporal pattern $P_1 = (C, N)\ b\ (G, VH)$ is an RTP in Z_i because $P_1 \in Z_i$, $(G, VH, 16, 23)$ is a recent state interval in Z_i, and $(C, N, 2, 14)$ is "close to" $(G, VH, 16, 23)$ $(16-14 \leq g)$. On the other hand, $P_2 = (G, H)\ b\ (G, N)$ is not an RTP in Z_i because $(G, N, 6, 9)$ is not a recent state interval.*

Definition 5. *(**Suffix**) Given temporal patterns $P = (\langle S_1, \ldots, S_{k_1} \rangle, R)$ and $P' = \left(\left\langle S'_1, \ldots, S'_{k_2} \right\rangle, R'\right)$ with $k_1 \leq k_2$, we say that P is a* ***suffix subpattern*** *of P', denoted as* ***Suffix(P, P′)****, if:*

$$\forall i \in \{1, \ldots, k_1\} \wedge j \in \{i+1, \ldots, k_1\} : S_i = S'_{i+k2-k1} \wedge R_{i,j} = R'_{i+k2-k1, j+k2-k1}$$

If P is a suffix subpattern of P', we say that P' is a **backward-extension superpattern** of P.

Proposition 1. *Given an MSS Z and temporal patterns P and P′, $R_g(P', Z) \wedge Suffix(P, P') \Rightarrow R_g(P, Z)$*

The proof directly follows from Definition 4.

Example 3. *Assume that $P = (\langle S_1, S_2, S_3\rangle, R_{1,2}, R_{1,3}, R_{2,3})$ is an RTP in Z. Proposition 1 says that its suffix subpattern $(\langle S_2, S_3\rangle, R_{2,3})$ must also be an RTP in Z. However, this does not imply that $(\langle S_1, S_2\rangle, R_{1,2})$ must be an RTP (the second condition of Definition 4 may be violated) nor that $(\langle S_1, S_3\rangle, R_{1,3})$ must be an RTP (the third condition of Definition 4 may be violated).*

Definition 6. ***(Frequent RTP)** Given a dataset D of MSS, a maximum gap parameter g and a minimum support threshold σ, we define the support of an RTP P as **$RTP\text{-}sup_g(P, D)$** = $|\{Z_i : Z_i \in D \wedge R_g(P, Z_i)\}|$. We say that P is a **frequent RTP** in D given σ if $RTP\text{-}sup_g(P, D) \geq \sigma$.*

Note that Proposition 1 implies the following property of *RTP-sup*, which we will use in our algorithm for mining frequent *RTPs*.

$$\forall P, P' \in TP, Suffix\left(P, P'\right) \Rightarrow RTP\text{-}sup_g\left(P, D\right) \geq RTP\text{-}sup_g\left(P', D\right)$$

4.2 The Mining Algorithm

In this section, we present the algorithm for mining frequent *RTPs*. We chose to utilize the class information and *mine frequent RTPs from each class label separately using local minimum supports* as opposed to mining frequent *RTPs* from the entire data using a single global minimum support. The approach is reasonable when pattern mining is applied in the supervised setting because 1) for unbalanced data, mining frequent patterns using a global minimum support threshold may result in missing many important patterns in the rare classes and 2) mining patterns that are frequent in one of the classes (hence potentially predictive for that class) is more efficient than mining patterns that are globally frequent.

The algorithm takes as input D_y: the MSS from class y, g: the maximum gap parameter and σ_y: the local minimum support threshold for class y. It outputs all temporal patterns that satisfy:

$$\{P \in TP : RTP - sup_g\left(P, D_y\right) \geq \sigma_y\}$$

The mining algorithm performs a level-wise search. It first scans the database to find all frequent *1-RTPs* (recent states). Then it extends the patterns ***backward in time*** to find more

complex temporal patterns. For each level *k*, the algorithm performs the following two phases to obtain the frequent *(k+1)-RTPs*:

1. **The candidate generation phase:** Generate candidate *(k+1)-patterns* by extending frequent *k-RTPs* backward in time.
2. **The counting phase:** Obtain the frequent *(k+1)-RTPs* by removing the candidates with *RTP-sup* less than σ_y.

This process repeats until no more frequent *RTPs* can be found.

In the following, we describe in details the candidate generation algorithm. Then we proposed techniques to improve the efficiency of candidate generation and counting.

4.2.1 Backward Candidate Generation—We generate a candidate *(k+1)-pattern* by appending a new state (*1-pattern*) to the beginning of a frequent *k-RTP*. Let us assume that we are backward extending pattern $P = (\langle S_1, \ldots, S_k \rangle, R)$ with state S_{new} to generate candidates of the form $\left(\left\langle S'_1, \ldots, S'_{k+1} \right\rangle, R'\right)$. First of all, we set $S'_1 = S_{new}$, $S'_{i+1} = S_i$ for $i \in \{1, \ldots, k\}$ and $R'_{i+1,j+1} = R_{i,j}$ for $i \in \{1, \ldots, k-1\} \wedge j \in \{i+1, \ldots, k\}$. This way, we know that every candidate P' of this form is a *backward-extension superpattern* of *P*: *Suffix*(*P*, *P*′).

In order to fully define a candidate, we still need to specify the temporal relations between the new state S'_1 and states $S'_2, \ldots, S'_{k+1}$, i.e., we should define $R'_{1,i}$ for $i \in \{2, \ldots, k+1\}$. Since we have two possible temporal relations (*before* and *co-occurs*), there are 2^k possible ways to specify the missing relations, resulting in 2^k different candidates. Let *L* denote all possible states and let F_k denote all frequent *k-RTPs*, generating the *(k+1)-candidates* **naively** in this fashion results in $2^k \times |L| \times |F_k|$ different candidates.

This large number of candidates makes the mining algorithm computationally very expensive and limits its scalability. Below, we describe the concept of incoherent patterns and introduce a method that generates fewer candidates without missing any real pattern from the mining results.

4.2.2 Improving the Efficiency of Candidate Generation—Definition 7. *A temporal pattern P is* ***incoherent*** *if there does not exist any valid MSS that contains P.*

Clearly, we do not have to generate and count incoherent candidates because we know that they will have zero support in the data. We introduce the following two lemmas to avoid generating incoherent candidates when specifying the relations $R'_{1,i}: i \in \{2, \ldots, k+1\}$ in candidates of the form $P' = \left(\left\langle S'_1, \ldots, S'_{k+1} \right\rangle, R'\right)$.

Lemma 1. $P' = \left(\left\langle S'_1, \ldots, S'_{k+1} \right\rangle, R'\right)$ *is incoherent if* $\exists i \in \{2, \ldots, k+1\} : R'_{1,i} = c$ *and* $S'_1.F = S'_i.F$.

Two state intervals from the same temporal variable cannot co-occur because temporal abstraction segments each variable into non-overlapping state intervals.

LEMMA 2. $P' = \left(\left\langle S'_1, \ldots, S'_{k+1} \right\rangle, R'\right)$ *is incoherent if* $\exists i \in \{2, \ldots, k+1\} : R'_{1,i} = c \wedge \exists j \in \{2, \ldots, i-1\} : R'_{1,j} = b$.

PROOF. Let us assume that there exists an MSS $Z = \langle E_1, \ldots, E_l \rangle$ where $P' \in Z$. Let π be the mapping from the states of P' to the state intervals of Z. The definition of temporal patterns and the fact that state intervals in Z are ordered by their start values implies that the matching state intervals $\langle E_{\pi(1)}, \ldots, E_{\pi(k+1)} \rangle$ are also ordered by their start times: $E_{\pi(1)}.s \leq \ldots \leq E_{\pi(k+1)}.s$. Hence, $E_{\pi(j)}.s \leq E_{\pi(i)}.s$ since $j < i$. We also know that $E_{\pi(1)}.e < E_{\pi(j)}.s$ because $R'_{1,j} = b$. Therefore, $E_{\pi(1)}.e < E_{\pi(i)}.s$. However, since $R'_{1,i} = c$, then $E_{\pi(1)}.e \geq E_{\pi(i)}.s$, which is a contradiction. Therefore, there is no MSS that contains P'.

EXAMPLE 4. *Assume we want to extend* $P = (\langle S_1 = (C, H), S_2 = (G, N), S_3 = (B, H), S_4 = (G, H) \rangle, R)$ *in Figure 4 with state* $S_{new} = (G, H)$ *to generate candidates of the form* $\left(\left\langle S'_1 = (G, H), S'_2 = (C, H), S'_3 = (G, N), S'_4 = (B, H), S'_5 = (G, H) \right\rangle, R'\right)$. *The relation between* S'_1 *and* S'_2 *is allowed to be either before or co-occurs:* $R'_{1,2} = b$ *or* $R'_{1,2} = c$. *However, according to Lemma 1,* $R'_{1,3} \neq c$ *because both* S'_1 *and* S'_3 *belong to the same temporal variable (G), which in turn implies that* $R'_{1,4} \neq c$ *and* $R'_{1,5} \neq c$ *according to Lemma 2. By removing incoherent patterns, we reduce the number of candidates that result from adding* (G, H) *to 4-RTP P from* $2^4 = 16$ *to only 2.*

THEOREM 1. *There are at most* $k + 1$ *coherent candidates that result from backward extending a single k-RTP with a new state.*

PROOF. We know that every candidate $P' = \left(\left\langle S'_1, \ldots, S'_{k+1} \right\rangle, R'\right)$ corresponds to a specific assignment of $R'_{1,i} \in \{b, c\}$ for $i \in \{2, \ldots k+1\}$. When we assign the temporal relations, once a relation becomes *before*, all the following relations have to be *before* as well according to Lemma 2. We can see that the relations can be *co-occurs* in the beginning of the pattern, but once we see a *before* relation at point $q \in \{2, \ldots, k+1\}$ in the pattern, all subsequent relations ($i > q$) should be *before* as well:

$$R'_{1,i} = c : i \in \{2, \ldots, q-1\}; \quad R'_{1,i} = b : i \in \{q, \ldots, k+1\}$$

Therefore, the total number of coherent candidates cannot be more than $k + 1$, which is the total number of different combinations of consecutive *co-occurs* relations followed by consecutive *before* relations.

In some cases, the number of coherent candidates is less than $k + 1$. Assume that there are some states in P' that belong to the same variable as state S'_1. Let S'_j be the first such state (j

$\leq k+1$). According to Lemma 1, $R'_{1,j} \neq c$. In this case, the number of coherent candidates is $j-1 < k+1$.

Algorithm 1 illustrates how to extend a *k-RTP P* with a new state S_{new} to generate coherent candidates (without violating Lemmas 1 and 2).

ALGORITHM 1

Extend backward a *k-RTP P* with a state S_{new}.

Input: A *k-RTP*: $P = (\langle S_1, ..., S_k \rangle, R)$; a new state: S_{new}
Output: Coherent candidates: C

$S'_1 = S_{new}$; $S'_{i+1} = S_i : i \in \{1, ..., k\}$;
$R'_{i+1,j+1} = R_{i,j} : i \in \{1, ..., k-1\}, j \in \{i+1, ..., k\}$;
$R'_{1,i} = b : i \in \{2, ..., k+1\}$; $P' = (\langle S'_1, ..., S'_{k+1} \rangle, R')$;
$C = \{P'\}$;
for *i=2* **to** *k+1* **do**
  if $(S'_1.F = S'_i.F)$ **then**
    break;
  else
    $R'_{1,i} = c$; $P' = (\langle S'_1, ..., S'_{k+1} \rangle, R')$;
    $C = C \cup \{P'\}$;
  end
end
return C

Corollary 1. *Let L denote all possible states and let F_k denote all frequent k-RTPs. The number of coherent (k+1)-candidates is always less or equal to* $(k+1) \times |L| \times |F_k|$.

4.2.3 Improving the Efficiency of Counting—Even after eliminating incoherent patterns, the mining algorithm is still computationally expensive because for every candidate, we need to scan the *entire database* in the counting phase to determine its *RTP-sup*. The question we try to answer in this section is whether we can omit portions of the database that are guaranteed not to contain the candidate we want to count. The proposed solution is inspired by [32] that introduced the *vertical format* for itemset mining and later applied it for sequential pattern mining [31].

Let us associate every frequent *RTP P* with a list of identifiers for all MSS that have *P* as an *RTP* (Definition 4):

$$P.RTP-list = \langle i_1, i_2, \ldots, i_n \rangle : Z_{i_j} \in D_y \wedge R_g\left(P, Z_{i_j}\right)$$

Clearly, $RTP\text{-}sup_g(P, D_y) = |P.RTP\text{-}list|$.

Let us also associate every state *S* with a list of identifiers for all MSS that contain *S* (Definition 2):

$$S.list = \langle q_1, q_2, \ldots, q_m \rangle : Z_{q_j} \in D_y \wedge S \in Z_{q_j}$$

Now, when we generate candidate P' by backward extending *RTP P* with state *S*, we define the potential list (***p-RTP-list***) of P' as follows:

$$P'.p-RTP-list = P.RTP-list \cap S.list$$

Proposition 2. *Let P' be a backward-extension of RTP P with state S: P'.RTP-list $\subseteq$ P'.p-RTP-list*

Proof. Assume Z_i is an MSS such that $R_g(P', Z_i)$. By definition, $i \in P'$.*RTP-list*. We know that $R_g(P', Z_i) \Rightarrow P' \in Z_i \Rightarrow S \in Z_i \Rightarrow i \in S.list$. Also, we know that *Suffix*(P, P') (Definition 5) $\Rightarrow R_g(P, Z_i)$ (Proposition 1) $\Rightarrow i \in P.RTP\text{-}list$. Therefore, $i \in P.RTP\text{-}list \cap S.list = P'.p\text{-}RTP\text{-}list$

Putting it all together, we compute the *RTP-lists* in the counting phase (based on the true matches) and the *p-RTP-lists* in the candidate generation phase. The key idea is that when we count candidate P', we only need to check the instances in its *p-RTP-list* because according to Proposition 2: $i \notin P'.p\text{-}RTP\text{-}list \Rightarrow i \notin P'.RTP\text{-}list \Rightarrow P'$ is not an *RTP* in Z_i. This offers a lot of computational savings because the *p-RTP-lists* get smaller as the size of the patterns increases, making the counting phase much faster.

Algorithm 2 outlines the candidate generation. Line 4 generates coherent candidates using Algorithm 1. Line 6 computes the *p-RTP-list* for each candidate. Note that the cost of the intersection is *linear* because the lists are always sorted according to the order of the instances in the database. Line 7 applies an *additional pruning* to remove candidates that are guaranteed not to be frequent according to the following implication of Proposition 2:

$$|P'.p-RTP-list| < \sigma_y \Rightarrow |P'.RTP-list| = RTP-sup_g(P, D_y) < \sigma_y$$

ALGORITHM 2

A high-level description of candidate generation.

Input: All frequent *k-RTPs*: F_k; all frequent states: L
Output: Candidate *(k+1)-patterns*: $Cand$, with their *p-RTP-lists*

$Cand = \Phi$;
foreach $P \in F_k$ **do**
  foreach $S \in L$ **do**
    C = *extend_backward*(P, S); (Algorithm 1)
    for $q = 1$ *to* $|\ C\ |$ **do**
      $C[q]$.*p-RTP-list* $= P$.*RTP-list* $\cap\ S.list$;
      if $(\ |\ C[q]$.*p-RTP-list* $|\ \geq \sigma_y\ )$ **then**
        $Cand = Cand \cup \{C[q]\}$;
      end
    end
  end
end
return $Cand$

4.3 Learning the Classifier

In this section, we summarize our approach for learning classification models for event detection problems. Given a training dataset $\{<x_i, y_i>\}_{i=1}^{n}$, where x_i is a multivariate time series instance up to time t_i and y_i is a class label at t_i, apply the following steps:

1. Convert every instance x_i to an MSS Z_i using temporal abstraction.
2. Mine the frequent *RTPs* from the MSS of each class label separately and combine the class-specific *RTPs* to obtain the final result Ω.
3. Convert every MSS Z_i into a binary vector $x_i^{'}$ of size equal to $|\Omega|$, where $x_{i,j}^{'}$ corresponds to a specific pattern $P_j \in \Omega$ and its value is 1 if $R_g(P_j, Z_i)$; and 0 otherwise.
4. Learn the classification model on the transformed binary representation of the training data $\left\{<x_i^{'}, y_i>\right\}_{i=1}^{n}$.

5. EXPERIMENTAL EVALUATION

In this section, we present our experiments on large-scale electronic health record (EHR) data collected for diabetic patients. We test our approach on the problem of detecting various types of disorders that are frequently associated with diabetes.

5.1 Dataset

The diabetes dataset consists of 13,558 records of adult diabetic patients (both type I and type II diabetes). Each patient's record consists of time series of 19 different lab values, including blood glucose, creatinine, glycosylated hemoglobin, blood urea nitrogen, liver function tests, cholesterol, etc. In addition, we have access to time series of ICD-9 diagnosis

codes reflecting the diagnoses made for the patient over time. Overall, the database contains 602 different ICD-9 codes. These codes were grouped by our medical expert into nine major categories: cardiovascular disease, renal disease, peripheral vascular disease, neurological disease, metabolic disease, inflammatory disease, ocular disease, cerebrovascular disease and hypertension. These disease categories are frequently associated with diabetes. Our objective is to learn models that are able to accurately diagnose these diseases. More specifically, at any point in time, we are interested in assigning a label for the disorder the patient with the diabetes suffers from. We omit the hypertension category from the analysis because it occurred in almost all patients, making it difficult to find negative examples.

5.2 Experimental Setup

The experiments are performed separately for each of the 8 major diagnosis categories (diseases). For each category, we divide the data into cases (positives) and controls (negatives) as follows:

- The *cases* are records of patients with the target disease that include clinical variables up to the time the disease was *first* diagnosed.
- The *controls* are selected randomly from the remaining patients (without the target disease) and they include clinical variables up to a randomly selected time point in the patient's record.

To avoid having uninformative training data, we discard instances that contain less than 10 lab measurements or that span less than 3 months (short instances). We choose the same number of controls as the number of cases for each category to make the datasets balanced. Table 1 shows the number of cases for each diagnosis category (the number of controls is the same).

To construct the features, we consider both the laboratory tests and the diagnosis codes. Note that the diagnosis of one or more disease categories may be predictive of the (first) occurrence of another disease, so it is important to include them as features. Laboratory tests are represented as numeric time series. We abstract them using value abstraction (see Section 3.1). Diagnosis categories, when used as features, are represented as intervals that start at the time of the diagnosis and extend until the end of the record.

5.3 Classification Performance

In this section, we test the ability of our *RTP* mining framework to represent and capture temporal patterns important for the prediction task. In particular, we compare the classification performance of the following feature construction methods:

1. ***Last_values:*** The features are formed from the most recent values of each clinical variable[6].
2. ***TP:*** The features correspond to all frequent temporal patterns.

[6]The features are numeric for the laboratory variables (e.g., last creatinine value is 2.2) and binary for the disease categories (whether or not the disease was diagnosed).

3. ***TP_sparse:*** The features correspond to the top 50 discriminative temporal patterns that are selected using a sparse linear model. These features are obtained by adjusting the cost parameter of an L_1 regularized support vector machine (SVM) classifier until at most 50 patterns are used for classification (features corresponding to other patterns are zeroed out).
4. ***RTP:*** The features correspond to all frequent *RTPs*.
5. ***RTP_sparse:*** The features correspond to the top 50 discriminative *RTPs* that are selected using a sparse linear model (similar to 3).

The first method is atemporal and only considers the most recent values for defining the classification features (a static transformation). On the other hand, methods (2-5) use temporal patterns (built using temporal abstractions and temporal relations) as their features (a dynamic transformation). For *TP* (*TP_sparse*), the feature value is one if the corresponding temporal pattern occurs anywhere in the instance (Definition 2), and zero otherwise. For *RTP* (*RTP_sparse*), the feature value is one if the corresponding temporal pattern occurs recently in the instance (Definition 3), and zero otherwise.

For methods (2-5), we set the local minimum supports (σ_y) to 15% of the number of instances in the class. For the *RTP* mining methods (4-5), we set the maximum gap parameter (g) to 6 months. The reason for including methods 3 and 5 is to test the ability of *TP* and *RTP* to represent the target disease using only a limited number of temporal patterns (50 patterns in our case).

We judged the quality of the different feature representations in terms of their induced classification performance. More specifically, we use the features extracted by each method to build a linear **SVM** classifier and evaluate its performance using the classification accuracy and the area under the ROC curve (AUC). We did not compare against other time series classification methods because most methods [24, 6, 28, 4] cannot be directly applied on multivariate irregularly sampled time series data.

Below, we show the classification accuracy (Table 2) and the AUC (Table 3) for each feature representation method on each classification task (major disease). All classification results are reported using averages obtained via *10-folds cross validation*.

The results show that features based on temporal patterns are beneficial for the classification task, since they outperform features based on most recent values (see for example the *NEURO*, *OCLUR* and *CARDI* datasets). The results also show that *RTP* and *RTP_sparse* mostly outperform *TP* and *TP_sparse*. It is important to note that although patterns generated by *TP* subsume the ones generated by *RTP* (by definition, every frequent *RTP* is also a frequent *TP*), the induced binary features are often different. For instance, a pattern that is not discriminative when considered in the entire records may become more discriminative when considered as a recent pattern. This can be seen clearly for the *RENAL* dataset, where *TP* and *TP_sparse* perform poorly because the discriminative signal is mostly contained in the recent values.

5.4 Knowledge Discovery

Figure 5 compares the number of temporal patterns that are extracted by frequent temporal pattern mining (*TP*) and by frequent *RTP* mining (*RTP*). Similar to the previous setting, we set the local minimum supports for both methods to 15% and we set the maximum gap parameter for *RTP* to 6 months.

The results show that the number of temporal patterns mined by *RTP* is at least an order of magnitude smaller than the number of patterns mined by *TP* for all datasets. This can facilitate the process of reviewing and validating the patterns by human experts.

Table 4 shows some of the top predictive *RTPs* according to their precision (confidence)[7]. The first three *RTPs* (P_1, P_2 and P_3) are predicting renal (kidney) disease. These patterns relate the risk of renal problems with very high values of the BUN test (P_1), an increase in creatinine levels from normal to high (P_2), and high values of BUN co-occurring with high values of creatinine (P_3). P_4 shows that an increase in glucose levels from high to very high may indicate a metabolic disease. Finally, P_5 indicates that patients who were previously diagnosed with cardiovascular disease and exhibit an increase in glucose levels are prone to develop a cerebrovascular disease. These patterns, extracted automatically from data without incorporating prior clinical knowledge, are in accordance with the medical diagnosis guidelines.

5.5 Mining Efficiency

In this section, we study the efficiency of different temporal pattern mining methods. In particular, we compare the running time of the following methods:

1. ***TP_Apriori:*** Mine frequent temporal patterns by extending the Apriori algorithm [1, 2] to the time interval domain. This method applies the Apriori pruning in the candidate generation phase to prune any candidate *k-pattern* if it contains an infrequent *(k-1)-patterns*.
2. ***RTP_no-lists:*** Mine frequent *RTPs* backward in time as described in this paper. However, this method does not apply the technique we propose in Section 4.2.3 to speed up the counting phase. This means that it scans the entire dataset for each candidate in order to compute its *RTP-sup*.
3. ***TP_lists:*** Mine frequent temporal patterns by extending the vertical format [32, 31] to the time interval domain as described in [5]. This method applies the Apriori pruning [1] in candidate generation and use id-lists to speed up the counting.
4. ***RTP_lists:*** Our proposed method for mining frequent *RTPs*.

To make the comparison fair, all methods apply the techniques we propose in Section 4.2.2 to avoid generating incoherent candidates. Note that if we do not remove incoherent candidates, the execution time for all methods greatly increases.

[7]Most of the highest precision *RTPs* are predicting the RENAL category because it is the easiest prediction task. So to diversify the patterns, we show the top 3 predictive *RTPs* for RENAL and the top 2 predictive *RTPs* for other categories.

The experiments are conducted on a Dell PowerEdge R610 server with an Intel Xeon 3.3GHz CPU and 96GB of RAM. Similar to the previous settings, we set the local minimum supports to 15% and the maximum gap parameter to 6 months (unless stated otherwise).

Figure 6 shows the execution time (in seconds) of the above methods on all major diagnosis datasets. We can see that our proposed *RTP_lists* method is much more efficient the other methods. For instance, on the *INFLM* dataset, *RTP_lists* is around 5 times faster than *TP_lists*, 10 times faster than *RTP_no-lists* and 30 times faster than *TP_Apriori*.

Figure 7 compares the execution time (in seconds) of the methods on the *CARDI* dataset for different minimum support thresholds. Note that the difference in the execution time between *RTP_lists* and the other methods becomes larger when the minimum support is low (10%).

Finally, let us examine the effect of the maximum gap parameter (g) on the efficiency of recent temporal pattern mining. Figure 8 shows the execution time (in seconds) of all methods on the *CARDI* dataset for different values of g (the execution time of *TP_Apriori* and *TP_lists* does not depend of g).

Clearly, the execution time of both *RTP_no-lists* and *RTP_lists* increases with g because the search space becomes larger (more temporal patterns become *RTPs*). The figure shows that when the maximum gap is 21 months, *RTP_no-lists* becomes slower than *TP_Apriori*. The reason is that for large values of g, applying the Apriori pruning [1] in candidate generation becomes more efficient (generates less candidates) than the backward extension of temporal patterns (see Example 3). On the other hand, *RTP_lists* increases much slower with g and maintains its efficiency advantage over *TP_lists* for larger values of g.

6. CONCLUSION

The increasing availability of large temporal datasets prompts the development of scalable and more efficient temporal pattern mining techniques. Methods for mining sequential (time-point) data were first introduced in the literature starting in the mid-1990 [2, 31, 18, 30]. Since then, these methods have been extended to mining time interval data [22, 11, 17, 14, 26, 27, 15]. Unfortunately, mining the entire set of temporal patterns (sequential patterns or time-interval patterns) from large-scale datasets is inherently a computationally expensive task. To alleviate this problem, temporal constraints (e.g., restricting the total pattern duration or restricting the permitted gap between consecutive events in a pattern) have been proposed to scale up the mining [19].

In this paper, we proposed a new class of temporal constraints for finding Recent Temporal Patterns (*RTPs*), which are particularly important for monitoring and event detection problems. We presented an efficient algorithm that mines time-interval patterns backward in time, starting from patterns related to the most recent observations. Our experimental evaluation on EHRs for diabetes patients showed that the *RTP* framework is very useful to efficiently find patterns that are important for predicting various types of disorders associated with diabetes.

ACKNOWLEDGMENTS

This work was supported by grants 1R21LM009102-01A1, 1R01LM010019-01A1, 1R01GM088224-01 and T15LM007059-24 from the NIH. Its content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH.

REFERENCES

[1]. Agrawal, R.; Srikant, R. Fast algorithms for mining association rules in large databases; Proceedings of VLDB; 1994;

[2]. Agrawal, R.; Srikant, R. Mining sequential patterns; Proceedings of ICDE; 1995;

[3]. Allen F. Towards a general theory of action and time. Artificial Intelligence. 1984; 23:123–154.

[4]. Batal, I.; Hauskrecht, M. A supervised time series feature extraction technique using dct and dwt; Proceedings of ICMLA; 2009;

[5]. Batal, I.; Valizadegan, H.; Cooper, GF.; Hauskrecht, M. A pattern mining approach for classifying multivariate temporal data; Proceedings of the IEEE international conference on bioinformatics and biomedicine (BIBM); 2011;

[6]. Blasiak, S.; Rangwala, H. A hidden markov model variant for sequence classification; Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI); 2011;

[7]. Chandola, V.; Eilertson, E.; Ertoz, L.; Simon, G.; Kumar, V. Data Warehousing and Data Mining Techniques for Computer Security. Springer; 2006. chapter:Data Mining for Cyber Security

[8]. Exarchos TP, Tsipouras MG, Papaloukas C, Fotiadis DI. A two-stage methodology for sequence classification based on sequential pattern mining and optimization. Data and Knowledge Engineering. 2008; 66:467–487.

[9]. Guttormsson S, Marks I, El-Sharkawi M, Kerszenbaum I. Elliptical novelty grouping for on-line short-turn detection of excited running rotors. IEEE Transactions on Energy Conversion. 1999 R.J.

[10]. Hauskrecht, M.; Valko, M.; Batal, I.; Clermont, G.; Visweswaram, S.; Cooper, G. Conditional outlier detection for clinical alerting; Proceedings of AMIA; 2010;

[11]. Höppner, F. Knowledge Discovery from Sequential Data. Technical University Braunschweig; Germany: 2003. PhD thesis

[12]. Ifrim, G.; Wiuf, C. Bounded coordinate-descent for biological sequence classification in high dimensional predictor space; Proceedings of the international conference on Knowledge Discovery and Data mining (SIGKDD); 2011;

[13]. Li L, Prakash BA, Faloutsos C. Parsimonious linear fingerprinting for time series. PVLDB. 2010; 3:385–396.

[14]. Moerchen, F. Algorithms for time series knowledge mining; Proceedings of SIGKDD; 2006; p. 668-673.

[15]. Moskovitch, R.; Shahar, Y. Medical temporal-knowledge discovery via temporal abstraction; Proceedings of AMIA; 2009;

[16]. Papadimitriou, S.; Sun, J.; Faloutsos, C. Streaming pattern discovery in multiple time-series; Proceedings of VLDB; 2005;

[17]. Papapetrou, P.; Kollios, G.; Sclaroff, S. Discovering frequent arrangements of temporal intervals; Proceedings of ICDM; 2005;

[18]. Pei, J.; Han, J.; Mortazavi-asl, B.; Pinto, H.; Chen, Q.; Dayal, U.; chun Hsu, M. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth; Proceedings of the 17th International Conference on Data Engineering (ICDE); 2001;

[19]. Pei J, Han J, Wang W. Constraint-based sequential pattern mining: the pattern-growth methods. Journal of Intelligent Information Systems. 2007; 28:133–160.

[20]. Ratanamahatana, C.; Keogh, EJ. Three myths about dynamic time warping data mining; Proceedings of the SIAM International Conference on Data Mining (SDM); 2005;

[21]. Shahar Y. A Framework for Knowledge-Based Temporal Abstraction. Artificial Intelligence. 1997; 90:79–133.

[22]. shan Kam, P.; chee Fu, AW. Discovering temporal patterns for interval-based events; Proceedings of the DaWaK; 2000;

[23]. Srivastava A, Kundu A, Sural S, Majumdar A. Credit card fraud detection using hidden markov model. IEEE Transactions on Dependable and Secure Computing. 2008

[24]. Vail, DL.; Veloso, MM.; Lafferty, JD. Conditional random fields for activity recognition; Proceedings of the international joint conference on Autonomous agents and multiagent systems (AAMAS); 2007;

[25]. Valko, M.; Hauskrecht, M. Feature importance analysis for patient management decisions; Proceedings of MedInfo; 2010;

[26]. Winarko E, Roddick JF. Armada - an algorithm for discovering richer relative temporal association rules from interval-based data. Data and Knowledge Engineering. 2007; 63:76–90.

[27]. Wu S-Y, Chen Y-L. Mining nonambiguous temporal patterns for interval-based events. IEEE Transactions on Knowledge and Data Engineering. 2007; 19:742–758.

[28]. Xi, X.; Keogh, E.; Shelton, C.; Wei, L.; Ratanamahatana, CA. Fast time series classification using numerosity reduction; Proceedings of the International Conference on Machine Learning (ICML); 2006;

[29]. Yang, K.; Shahabi, C. A pca-based similarity measure for multivariate time series; Proceedings of the international workshop on Multimedia databases; 2004;

[30]. yen Lin M, yin Lee S. Fast discovery of sequential patterns through memory indexing and database partitioning. Journal Information Science and Engineering. 2005; 21:109–128.

[31]. Zaki M. Spade: An efficient algorithm for mining frequent sequences. Machine Learning. 2001; 42:31–60.

[32]. Zaki MJ. Scalable algorithms for association mining. IEEE Transactions on Knowledge and Data Engineering. 2000

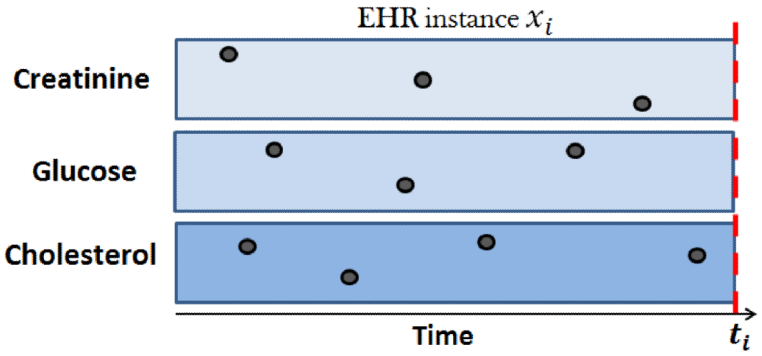


Figure 1. An example of an EHR data instance with three temporal variables. The black dots represent their values over time

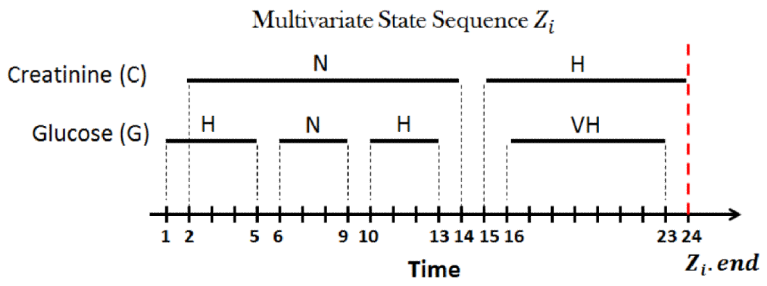


Figure 2. An MSS representing 24 days of a patient record. In this example, there are two temporal variables (creatinine and glucose)

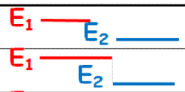
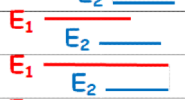
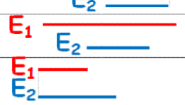
	E_1 before E_2	E_2 after E_1
	E_1 meets E_2	E_2 is-met-by E_1
	E_1 overlaps E_2	E_2 is-overlapped-by E_1
	E_1 is-finished-by E_2	E_2 finishes E_1
	E_1 contains E_2	E_2 during E_1
	E_1 starts E_2	E_2 is-started-by E_1
	E_1 equals E_2	E_2 equals E_1

Figure 3. Allen's temporal relations

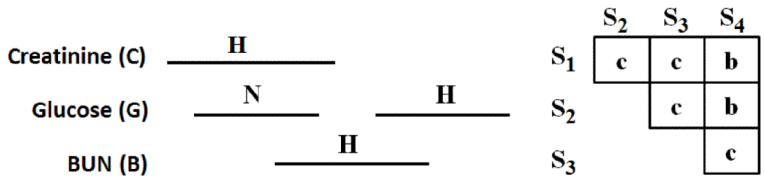


Figure 4. A temporal pattern with states $\langle (C, H), (G, N), (B, H), (G, H) \rangle$ and temporal relations $R_{1,2} = c$, $R_{1,3} = c$, $R_{1,4} = b$, $R_{2,3} = c$, $R_{2,4} = b$ and $R_{3,4} = c$

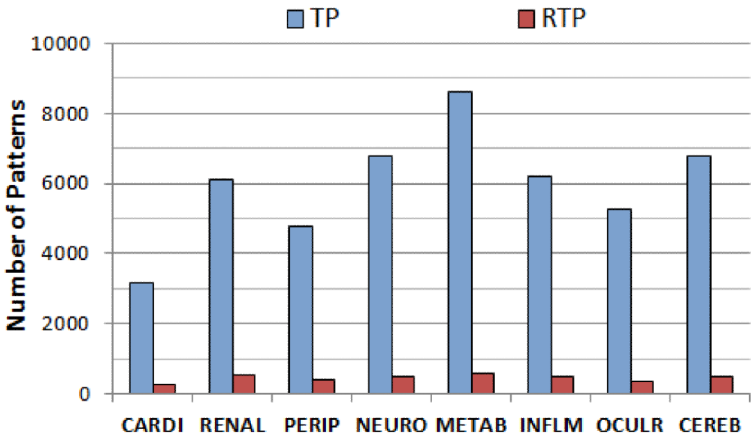


Figure 5. The number of temporal patterns of *TP* and *RTP* on all major diagnosis datasets (minimum support is 15%)

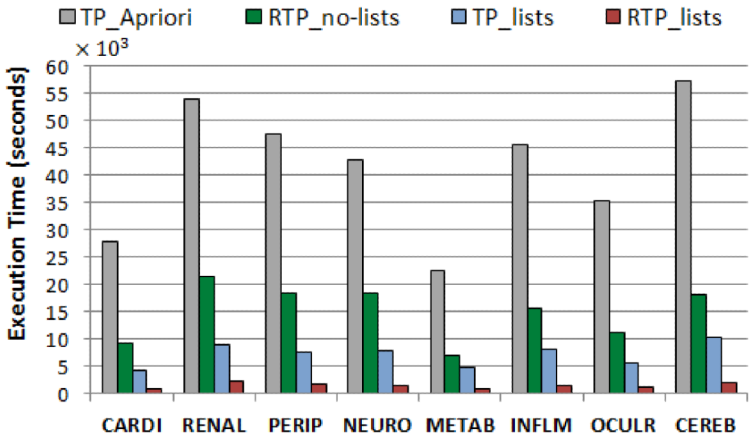


Figure 6. The mining time (in seconds) of *TP_Apriori*, *RTP_no-lists*, *TP_lists* and *RTP_lists* on all major diagnosis datasets (minimum support is 15%)

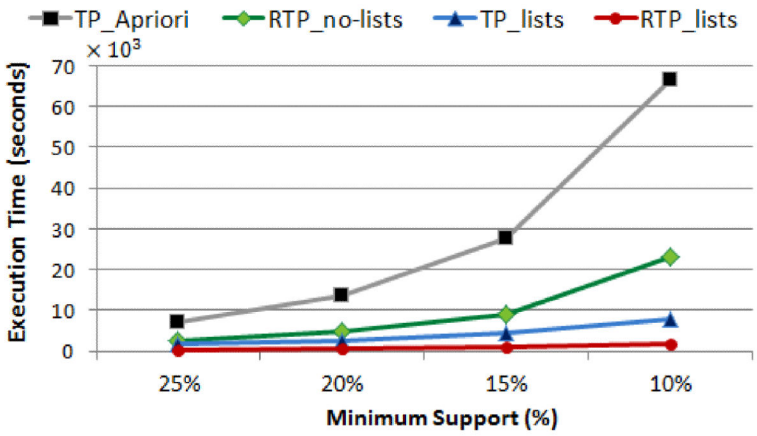


Figure 7. The mining time (in seconds) of *TP_Apriori*, *RTP_no-lists*, *TP_lists* and *RTP_lists* on the *CARDI* dataset for different minimum support values

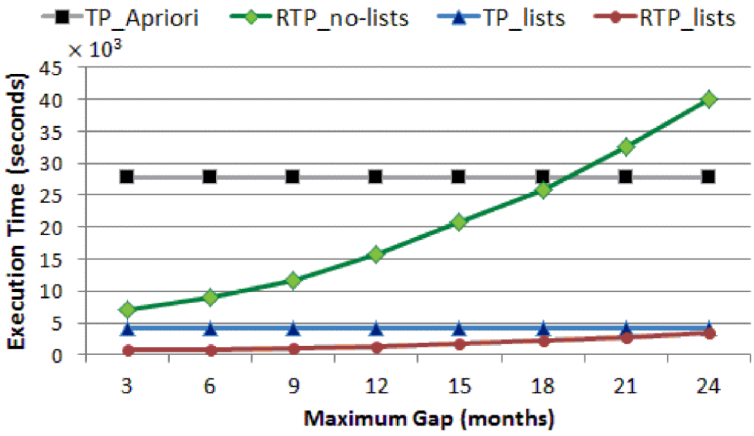


Figure 8. The mining time (in seconds) of *TP_Apriori*, *RTP_no-list*, *TP_lists* and *RTP_lists* on the *CARDI* dataset for different maximum gap values (in months). The minimum support is 15%

Table 1

The eight major diagnosis categories (diseases) used in the diabetes study and the number of cases for each category. The number of controls is set to be the same as the number of cases

Dx code	Description	# cases
CARDI	Cardiovascular disease	2,743
RENAL	Renal disease	3,355
PERIP	Peripheral vascular disease	3,370
NEURO	Neurological disease	2,193
METAB	Metabolic disease	968
INFLM	Inflammatory (infectious) disease	2,394
OCULR	Ocular (ophthalmologic) disease	2,245
CEREB	Cerebrovascular disease	2,824

Table 2

The classification accuracy for the different feature representation methods (SVM is used for classification)

Dataset	*Last_values*	*TP*	*TP_sparse*	*RTP*	*RTP_sparse*
CARDI	67.41	71.82	71.62	71.82	**71.98**
RENAL	77.71	76.38	76.66	78.08	**78.33**
PERIP	66.82	68.55	68.38	**70.01**	69.91
NEURO	64.66	68.95	68.33	69.18	**69.68**
METAB	72.83	**74.64**	73.61	73.3	73.09
INFLM	64.6	66.73	66.69	67.5	**67.94**
OCULR	65.83	**70.8**	70.71	68.82	69.22
CEREB	65.21	66.64	66.7	**67.93**	66.98

Table 3

The area under the ROC curve (AUC) for the different feature representationmethods (SVM is used for classification)

Dataset	*Last_values*	*TP*	*TP_sparse*	*RTP*	*RTP_sparse*
CARDI	75.1	80.13	79.61	80.18	**80.52**
RENAL	85.45	84.8	84.97	86.13	**86.23**
PERIP	74.38	76.08	75.95	77.88	**78.31**
NEURO	72.25	76.43	75.81	**77.34**	76.98
METAB	80.64	82.67	81.66	82.52	**82.97**
INFLM	71.04	73.62	73.43	74.39	**74.92**
OCULR	72.12	**78.34**	78.28	76.11	76.85
CEREB	72.23	73.46	74.42	**75.37**	75.18

Table 4

Predictive RTPs with their precision (*prec*) and recall

RTP	***prec***	***recall***
P_1: *BUN=VH* ⇒ *Dx=RENAL*	0.97	0.17
P_2: *Creat=N before Creat=H* ⇒ *Dx=RENAL*	0.96	0.21
P_3: *BUN=H co-occurs Creat=H* ⇒ *Dx=RENAL*	0.95	0.21
P_4: *Gluc=H before Gluc=VH* ⇒ *Dx=METAB*	0.79	0.24
P_5: *Dx=CARDI co-occurs* (*Gluc=N before Gluc=H*) ⇒ *Dx=CEREB*	0.71	0.22

Abbreviations: *Dx*: diagnosis code (one of the 8 major categories in Table 1**); *BUN*: Blood Urea Nitrogen; *Creat*: creatinine; *Gluc*: blood glucose. Abstractions: *BUN=VH*:**> **49 mg/dl; *BUN=H*:**> 34 **mg/dl; *Creat=H*:** > 1.8 **mg/dl; *Creat=N*:** [0.8-1.8] **mg/dl; *Gluc=VH*:** > 243 **mg/dl; *Gluc=H*:**>191 **mg/dl**.